Vignette: enaR

S.R. Borrett and M.K. Lau

October 3, 2012

Contents

1 Introduction **2**

2 Background **2**

3 Data Input: General **2**

3.1 Model Data 3

3.2 Network Data Class 3

3.3 Building a Network Object 4

3.4 Balancing for Steady-State 6

4 Data Input: Reading Common Data File Formats **7**

4.1 Reading a SCOR file 7

4.2 Reading in a WAND file 8

4.3 Reading in a ENAM file 8

5 Network Visualization **8**

6 Single Model Analysis **9**

6.1 Structural Network Analysis 9

6.2 Flow Analysis 11

6.3 Ascendency 14

6.4 Storage Analysis 14

6.5 Utility Analysis 15

6.6 Environ Analysis 15

6.7 Control Analysis 17

6.8 Mixed Trophic Impacts 17

6.9 Other Analyses 18

7 Multi-Model Analyses (Batch Processing) **21**

8 Connecting to Other Useful Packages **27**

8.1 sna: Social Network Analysis 27

8.2 iGraph 30

9 Summary and Future **31**

1 Introduction

This package is a collection of functions to implement Ecological Network Analysis (ENA), which is a family of algorithms for investigating the structure and function of ecosystems modeled as networks of thermodynamically conserved energy–matter exchanges. The package brings together multiple ENA algorithms from several approaches into one common software framework that is readily available and extensible. The package builds on the *network* data structure for R developed by Butts (2008*a*). In addition to being able to perform several types of ENA with a single package, users can also make use of network analysis tools built into the *network* package, the *sna* (social network analysis) package (Butts, 2008*b*), and other components of what is now called *statnet* (Handcock et al., 2008).

This vignette illustrates how to use the *enaR* package to perform ENA. It is not meant to be a detailed guide to ENA, but we provide some references to the primary literature for those wishing to learn more about the techniques.

2 Background

Before describing how to use this package, we provide a brief background of ENA. Users may find this helpful as several software design decisions were predicated on the history and current state of the field.

The ENA methodology is an application and extension of economic Input–Output Analysis (Leontief, 1936, 1966) that was first introduced into ecology by Hannon (1973). Two major schools have developed in ENA. The first is based on Dr. Robert E. Ulanowicz's work with a strong focus on trophic dynamics and a use of information theory (Ulanowicz, 1986, 1997, 2004). The second school has an environment focus and is built on the environ concept introduced by Dr. Bernard C. Patten (Fath and Patten, 1999; Patten, 1978; Patten et al., 1976). Patten's approach has been collectively referred to separately as *Network Environ Analysis*. At the core the two approaches are very similar; however, they make some different starting assumptions and follow independent yet braided development tracks. Recent work has started to bring the two approaches back together (e.g., Scharler and Fath, 2009). Borrett et al. (2012) provides an entry level overview of the field.

Disparate software packages have been created to support ENA. Ulanowicz first developed and distributed the DOS based NETWRK4 code, which is still available. Recently some of these algorithms were reimplemented in an Microsoft Excel based WAND package (Allesina and Bondavalli, 2004). Some of these methods have also been encoded in the popular Ecopath with Ecosim software that assists with model construction (Christensen and Walters, 2004). Fath and Borrett (2006) published NEA.m, a MATLAB©function that collected the Patten School's algorithms together into one set of code. One objective for this R package is to begin to bring together these different algorithms into a single accessible and extensible package.

3 Data Input: General

In this section we describe the data necessary for the Ecological Network Analysis and show how to build the central network data object in R that contains the model data for subsequent analysis. To start, we assume you have installed the enaR package, and then loaded the library as follows:

```
> library(enaR)
```

```
     Tools for Social Network Analysis
Version      2.2-0 created on      2010-11-21.
copyright (c) 2005, Carter T. Butts, University of California-Irvine
Type help(package="sna") to get started.

>
```

3.1 Model Data

ENA is applied to a network model of energy–matter exchanges among system components. The system is modeled as a set of n compartments or nodes that represent species, species-complexes (i.e., trophic guilds or functional groups), or non-living components of the system in which energy–matter is stored. Nodes are connected by L observed fluxes, termed directed edges or links. This analysis requires an estimate of the energy–matter flowing from node i to j over a given period, $\mathbf{F}_{n\times n} = [f_{ij}]$, $i, j = 1, 2, \ldots, n$. These fluxes can be generated by any process such as feeding (like a food web), excretion, and death. As ecosystems are thermodynamically open, there must also be energy–matter inputs into the system $\mathbf{z}_{1\times n} = [z_i]$, and output losses from the system $\mathbf{y}_{1\times n} = [y_i]$. While the Patten School treats all outputs the same, the Ulanowicz School typically partitions outputs into respiration $\mathbf{r}_{1\times n} = [r_i]$ and export $\mathbf{e}_{1\times n} = [e_i]$ to account for differences in energetic quality. Note that $y_i = r_i + e_i, \forall i$. Some analyses also require the amount of energy–matter stored in each node (e.g., biomass), $\mathbf{X}_{1\times n} = [x_i]$. The final required information is a categorization of each node as living or not, which is essential for algorithms from the Ulanowicz School. For our implementation, we have created a logical vector $\mathbf{Living}_{1\times n}$ that indicates whether the i^{th} node is living (TRUE) or not (FALSE). Together, the model data $\mathcal{M}$ can be summarized as $\mathcal{M} = \{\mathbf{F}, \mathbf{z}, \mathbf{e}, \mathbf{r}, \mathbf{X}, \mathbf{Living}\}$.

Notice the row-to-column orientation of $\mathbf{F}$. This is consistent with the Ulanowicz School of network analysis, as well as the orientation commonly used in Social Network Analysis and used in the *statnet* packages. However, this is the opposite orientation typically used in the Patten School of analysis that conceptually builds from a system of differential equations and thus uses the column-to-row orientation common in this area of mathematics. Even though the difference is only a matrix transpose, this single difference may be the source of much confusion in the literature and frustration on the part of users. We have selected to use row-to-column orientation for our primary data structure as it is the dominant form and it is the assumed orientation in the *statnet* packages we are building upon. To facilitate package use by the existing community, however, we have decided to return results in the orientation of the School from which the algorithm was taken. Thus, results of the Patten School algorithms are returned in their column-to-row orientation, while algorithms we have coded from the Ulanowicz school return results in the row-to-column orientation.

3.2 Network Data Class

The *enaR* package stores the model data in the **network** class defined in the *network* package (see Butts, 2008*a*, for details). Again, the primary network object components are:

- F = flow matrix oriented row to column
- z = inputs
- r = respiration
- e = exports

- y = respiration+exports
- X = biomass or storage values
- Living = logical vector indicating if the node is living (TRUE) or non-living (FALSE)

3.3 Building a Network Object

Users can assemble the necessary data elements described in Section 3.1 and then use the *pack* function to create the network data object. Here is an example of doing this with hypothetical data.

```
> # generate the flow matrix
> flow.mat <- array(abs(rnorm(100,4,2))*sample(c(0,1),100,replace=TRUE),
+                    dim=c(4,4))
> # name the nodes
> rownames(flow.mat) <- colnames(flow.mat) <- paste('node',(1:nrow(flow.mat)),sep='')
> # generate the inputs
> inputs <- runif(nrow(flow.mat),0,4)
> # generate the exports
> exports <- inputs
> # pack
> fake.model <- pack(flow=flow.mat,
+                     input=inputs,
+                     export=exports,
+                     living=TRUE)
> # model
> fake.model
```

```
 Network attributes:
  vertices = 4
  directed = TRUE
  hyper = FALSE
  loops = FALSE
  multiple = FALSE
  bipartite = FALSE
  flow:
     node1             node2            node3             node4
 Min.   :0.0000   Min.   :0.000   Min.   :0.000   Min.   :0.0000
 1st Qu.:0.0000   1st Qu.:1.789   1st Qu.:0.000   1st Qu.:0.0000
 Median :0.0000   Median :3.903   Median :2.190   Median :0.3489
 Mean   :0.5913   Mean   :3.369   Mean   :2.318   Mean   :1.0193
 3rd Qu.:0.5913   3rd Qu.:5.483   3rd Qu.:4.507   3rd Qu.:1.3682
 Max.   :2.3651   Max.   :5.672   Max.   :4.893   Max.   :3.3795
  balanced = FALSE
  total edges= 5
    missing edges= 0
    non-missing edges= 5
 Vertex attribute names:
    export input living output respiration storage vertex.names

>
```

Unfortunately, the attributes() function does not clearly identify the network data objects we are using.

```
> attributes(fake.model)

$names
[1] "mel" "gal" "val" "iel" "oel"

$class
[1] "network"
```

However, individual components can be extracted from the data object using the form specified in the *network* package. For example, we can pull out node of vertex attributes as follows

```
> fake.model%v%'output'

[1] 3.9472104 1.9428545 1.7257228 0.2475129

> fake.model%v%'input'

[1] 3.9472104 1.9428545 1.7257228 0.2475129

> fake.model%v%'living'

[1] TRUE TRUE TRUE TRUE
```

For convenience, we have defined the flow matrix as a network based characteristic and it can be extracted as

```
> fake.model%n%'flow'

         node1    node2    node3     node4
node1 0.000000 5.419827 0.000000 3.3795308
node2 0.000000 5.672006 4.892553 0.0000000
node3 2.365092 0.000000 4.379095 0.0000000
node4 0.000000 2.385203 0.000000 0.6978212
```

There are times that it is useful to extract all of the ecosystem model data elements from the network data object. This can be accomplished using the `unpack` function. The `unpack` output is as follows:

```
> unpack(fake.model)
```

```
$F
         node1    node2    node3     node4
node1 0.000000 5.419827 0.000000 3.3795308
node2 0.000000 5.672006 4.892553 0.0000000
node3 2.365092 0.000000 4.379095 0.0000000
node4 0.000000 2.385203 0.000000 0.6978212

$z
[1] 3.9472104 1.9428545 1.7257228 0.2475129

$r
[1] 0 0 0 0

$e
[1] 3.9472104 1.9428545 1.7257228 0.2475129

$y
[1] 3.9472104 1.9428545 1.7257228 0.2475129

$X
[1] 0 0 0 0

$Living
[1] TRUE TRUE TRUE TRUE

>
```

In this case we did not specify the node respiration in the model, so this vector is filled with zeros. Also, we did not specify a **Living** vector when we built the data object, so `pack` defaulted to the assumption that all the nodes were living.

3.4 Balancing for Steady-State

Many of the ENA functions assume that the network model is at steady-state (node inputs equal node outputs). Thus, this package has functions for (1) checking to see if the assumption is met and (2) automatically balancing the model so that input equal outputs.

To determine if the model is balanced and then balance it if necessary:

```
> ## --- Check to see if the model is balanced ---#
> ssCheck(fake.model)

[1] FALSE

> ## --- To BALANCE a model if needed --- #
> fake.model <- balance(fake.model,method="AVG2")

[1] AVG2

>
```

The automated balancing routines are based on those presented in Allesina and Bondavalli (2003). These authors compare alternative balancing algorithms and further discuss the implications of using automated procedures. Caution is warranted when using these techniques.

4 Data Input: Reading Common Data File Formats

Several software packages exist in the literature for running ENA. For convenience, we have written functions to read in a few of the more common data formats used by these software.

4.1 Reading a SCOR file

The *read.scor* function reads in data stored in the SCOR format specified by Ulanowicz and Kay (1991) that is the input to the NETWRK4 programs. This function can be run as follows.

```
> m <- read.scor("~/Dropbox/RENA/package/data/oyster.dat")
>
```

This constructs the network data object from the SCOR file that stores the ecosystem model data for an oyster reef model (Dame and Patten, 1981). The individual model elements are

```
> unpack(m)

$F
                  Filter Feeders Microbiota Meiofauna Deposit Feeders
Filter Feeders                 0     0.0000    0.0000          0.0000
Microbiota                     0     0.0000    1.2060          1.2060
Meiofauna                      0     0.0000    0.0000          0.6609
Deposit Feeders                0     0.0000    0.0000          0.0000
Predators                      0     0.0000    0.0000          0.0000
Deposited Detritus             0     8.1721    7.2745          0.6431
                  Predators Deposited Detritus
Filter Feeders       0.5135            15.7910
Microbiota           0.0000             0.0000
Meiofauna            0.0000             4.2403
Deposit Feeders      0.1721             1.9076
Predators            0.0000             0.3262
Deposited Detritus   0.0000             0.0000

$z
[1] 41.47  0.00  0.00  0.00  0.00  0.00

$r
[1] 25.1650  5.7600  3.5794  0.4303  0.3594  6.1759

$e
[1] 0 0 0 0 0 0

$y
[1] 25.1650  5.7600  3.5794  0.4303  0.3594  6.1759

$X
[1] 2000.0000    2.4121   24.1210   16.2740   69.2370 1000.0000
```

```
$Living
[1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
```

This same data is stored as a network data object that is distributed with this package, which can be accessed as:

```
> data(oyster)
> m <- oyster
```

4.2 Reading in a WAND file

In part to make ENA more accessible to biologists, Allesina and Bondavalli (2004) recoded some of Ulanowicz's NETWRK4 algorithms into a Microsoft Excel based tool called WAND. For this tool, the model data is stored as a separate Excel file with two worksheets. The first contains many of the node attributes and the second contains the flow matrix. The `read.wand` function will create an R network data object from a WAND model file.

```
> m <- read.wand(paste(system.file(package='enaR'),'data','MDmar02_WAND.xls',sep='/'))
```

This code creates a network data object for *enaR* from the WAND formatted Mdloti ecosystem model data (Scharler, 2012). This data is courtesy of U.M. Scharler.

4.3 Reading in a ENAM file

Another commonly used data format stores the necessary model data in a csv or Excel formatted file. We include an example Excel file of the Mdloti estuary stored in this form ("MDMAR02.xlsx", courtesy of U. M. Scharler). This format has not been described technically in the literature nor has it been named. We refer to it as ENAM as it is the ENA model data stored primarily as a square matrix with several preliminary rows that include meta-data, the number of nodes, and number of living nodes (similar to SCOR). The data format is generally similar in concept, if not exact form, to the data system matrix used as the input to the NEA.m function (Fath and Borrett, 2006). However, the ENAM format includes information on whether nodes are living and partitions output into respiration and exports.

This data format can be read into the *enaR* package as

```
> m <- read.enam(paste(system.file(package='enaR'),'data','MDMAR02.xlsx',sep='/'))
```

The current read.enam function assumes the data are stored on the first worksheet of an Excel file. In the future, we expect to expand this function's capabilities to read the data from a CSV file.

5 Network Visualization

The *enaR* package uses the *network* package plot tools. Here is one example of how to plot a network model. The figure scaling may need to be adjusted depending on computer and devices. Also note that the graph only shows internal system flows.

Figure 1 (left) is a very simple example of to plot a graph of the oyster reef model accomplished with default settings.

```
> data(oyster)  # load data
> m <- oyster
> set.seed(2)    # set random seed to control plot
> plot(m)       # plot network data object (uses plot.network)
```

We can use the excellent graphics capabilities of R to make fancier plot of the same data (Fig. 1(right)).

```
> # set colors to use
> my.col=c("red","yellow",
+    rgb(204,204,153,maxColorValue=255),
+    "grey22")
> F=m%n%'flow'                    # extract flow information for later use.
> f=which(F!=0, arr.ind=T)        # get indices of positive flows
> opar <- par(las=1,bg=my.col[4],xpd=TRUE,mai=c(1.02, 0.62, 0.82, 0.42))
> set.seed(2)                     # each time the plot is called, the
>                                 # layout orientation changes.  setting
>                                 # the seed ensures a consistent
>                                 # orientation each time the plot
>                                 # function is called.
> plot(m,
+       vertex.cex=log(m%v%'storage'), # scale nodes with storage
+       label= m%v%'vertex.names',     # add node labels
+       boxed.labels=FALSE,
+       label.cex=0.65,
+       vertex.sides=45,   # to make rounded
+       edge.lwd=log10(abs(F[f])),     # scale arrows to flow magnitude
+       edge.col=my.col[3],
+       vertex.col=my.col[1],
+       label.col="white",
+       vertex.border = my.col[3],
+       vertex.lty = 1,
+       xlim=c(-4,1),ylim=c(-2,-2))
> rm(opar)             # remove changes to the plotting parameters
>
```

6 Single Model Analysis

In practice, ENA is applied to a single model. Here, we walk through an example of applying multiple ENA algorithms to the oyster reef model (Dame and Patten, 1981).

6.1 Structural Network Analysis

Structural network analysis is common to many types of network analysis. The structural analyses applied here are based on those presented in NEA.m (Fath and Borrett, 2006) following the Patten School. Thus, the adjacency matrix returned is oriented from column-to-row. In addition, several network statistics are returned to describe the whole network. These include:

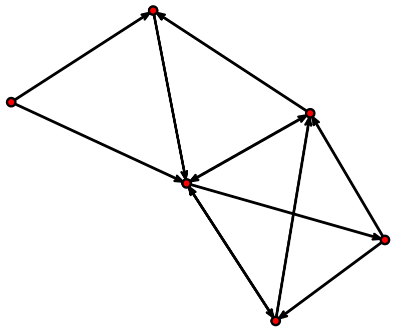
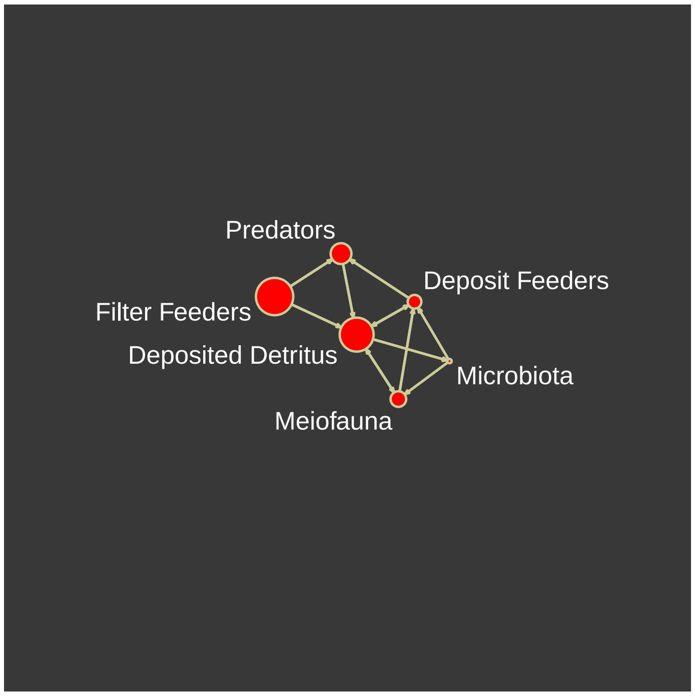

Figure 1: Simple (left) and fancy (right) plot of the Oyster network model (Dame and Patten 1981).

- n = number of nodes
- L = number of links
- C = connectance (L/n^2)
- LD = average link density L/n
- lam1A = dominant eigenvalue of A = rate of pathway proliferation (see Borrett et al., 2007, for details)
- mlam1A = multiplicity of the dominant eigenvalue
- lam2A = second largest eigenvalue
- rho = damping ratio = lam1A/lam2A indicates how fast the pathway proliferation rate reaches lam1A
- R = distance of lam1[A] from the bulk of the eigen spectrum (Farkas et al., 2001)
- d = difference between dominant eigenvalue and link density
- no.scc = number of strongly connected components (see Allesina et al., 2005; Borrett et al., 2007, for details on SCCs)
- no.scc.big = number of strongly connected components with n>1
- pscc = percent of nodes participating in a large scc

```
> St <- enaStructure(m)
> attributes(St)

$names
[1] "A"  "ns"

> St$ns

     n  L         C LD    lam1A mlam1A lam2A      rho         R
[1,] 6 12 0.3333333  2 2.147899      1     1 2.147899 0.4655712
            d no.scc no.scc.big      pscc
[1,] 0.147899      2          1 0.8333333

>
```

The structural network statistics show that the oyster reef model has 6 nodes, a pathway proliferation rate of 2.14, and that the model is comprised of two strongly connected components but that only one has more than one node.

6.2 Flow Analysis

Flow analysis or throughflow analysis is one of the core ENA analyses for both the Ulanowicz and Patten Schools. The *enaR* implementation `enaFlow` mostly folllows the NEA.m function, with small updates (e.g. calculating the ratio of indirect-to-direct flows Borrett and Freeze, 2011; Borrett et al., 2011). Thus, the resultant matrices are oriented column-to-row. Again, the function returns a number of whole network properties or network statistics. These include

- Boundary = Total boundary flow (input=output)
- TST = Total System ThroughFLOW
- APL = Average Path Length (Finn, 1976)
- FCI = Finn Cycling Index (Finn, 1980)
- BFI = Boundary Flow Intensity, Boundary/TST
- DFI = Direct Flow Intensity, Direct/TST
- IFI = Indirect Flow Intensity, Indirect/TST (Borrett et al., 2006)
- id = Ratio of Indirect to Direct Flow (realized = scaled to boundary flow, Borrett and Freeze (2011); Borrett et al. (2011))
- id.I = input oriented ratio of indirect to direct flow intensity ((as in Fath and Borrett, 2006))
- id.O = output oriented ratio of indirect to direct flow intensity ((as in Fath and Borrett, 2006))
- HMG.I = input oriented network homogenization to direct flow intensity
- HMG.O = output oriented network homogenization to direct flow intensity

- AMP.strong.I = input oriented network amplification
- AMP.strong.O = output oriented network amplification
- mode0 = Boundary Flow
- mode1 = internal First Passage Flow
- mode2 = Cycled Flow
- mode3 = dissipative equivalent to mode 1
- mode4 = dissipative equivalent to mode 0

Here, we extract the flow statistics and then isolate and remove the output-oriented direct flow intensity matrix **G** matrix. We also show the input-oriented integral flow matrix $\mathbf{N}'$.

```
>   F <- enaFlow(m)
> attributes(F)

$names
[1] "T"  "G"  "GP" "N"  "NP" "ns"

> F$ns

     Boundary     TST     TSTp      APL       FCI       BFI       DFI
[1,]    41.47 83.5833 125.0533 2.015512 0.1101686 0.4961517 0.1950689
           IFI       id     id.I     id.O    HMG.I    HMG.O
[1,] 0.3087794 1.582925 1.716607 1.534181 2.051826 1.891638
     AMP.strong.I AMP.strong.O mode0    mode1    mode2    mode3 mode4
[1,]            3            1 41.47 32.90504 9.208256 32.90504 41.47

> G <- F$G # output-oriented direct flow matrix
> rm(G)
> F$NP     # input-oriented integral flow matrix

                   Filter Feeders Microbiota Meiofauna Deposit Feeders
Filter Feeders                  1  0.0000000 0.0000000       0.0000000
Microbiota                      1  1.1018630 0.2971032       0.1240688
Meiofauna                       1  0.2440716 1.2971032       0.1240688
Deposit Feeders                 1  0.6197856 0.5604100       1.1240688
Predators                       1  0.1555792 0.1406747       0.2821649
Deposited Detritus              1  0.1018630 0.2971032       0.1240688
                   Predators Deposited Detritus
Filter Feeders     0.0000000          0.0000000
Microbiota         0.0203426          1.3885039
Meiofauna          0.0203426          1.3885039
Deposit Feeders    0.0203426          1.3885039
Predators          1.0051064          0.3485436
Deposited Detritus 0.0203426          1.3885039
```

Note: you can use the attach function to have access to the objects nested within an object. Since some objects may conflict in name, it's best to detach an object once it's not in use.

```
> attach(F)

The following object(s) are masked from 'package:base':

    T

> G

                   Filter Feeders Microbiota  Meiofauna Deposit Feeders
Filter Feeders         0.00000000  0.0000000 0.00000000      0.00000000
Microbiota             0.00000000  0.0000000 0.00000000      0.00000000
Meiofauna              0.00000000  0.1475753 0.00000000      0.00000000
Deposit Feeders        0.00000000  0.1475753 0.07793173      0.00000000
Predators              0.01238245  0.0000000 0.00000000      0.06856574
Deposited Detritus     0.38078129  0.0000000 0.50000590      0.76000000
                   Predators Deposited Detritus
Filter Feeders     0.0000000         0.00000000
Microbiota         0.0000000         0.36703630
Meiofauna          0.0000000         0.32672209
Deposit Feeders    0.0000000         0.02888377
Predators          0.0000000         0.00000000
Deposited Detritus 0.4757876         0.00000000

> detach(F)
>
```

Matrix powers – raising a matrix to a power is not a native operation in R. Thus, the *enaR* package includes a function `mExp` to facilitate this matrix operation commonly used in ENA.

```
> mExp(F$G,2)

                   Filter Feeders Microbiota   Meiofauna
Filter Feeders        0.000000000 0.00000000 0.000000000
Microbiota            0.139760556 0.00000000 0.183520316
Meiofauna             0.124409658 0.00000000 0.163362971
Deposit Feeders       0.010998399 0.01150080 0.014442055
Predators             0.000000000 0.01011861 0.005343446
Deposited Detritus    0.005891414 0.18594573 0.059228112
                   Deposit Feeders  Predators Deposited Detritus
Filter Feeders          0.00000000 0.00000000        0.000000000
Microbiota              0.27894759 0.17463133        0.000000000
Meiofauna               0.24830879 0.15545033        0.054165488
Deposit Feeders         0.02195166 0.01374254        0.079627504
Predators               0.00000000 0.00000000        0.001980437
Deposited Detritus      0.03262273 0.00000000        0.185314635

>
```

6.3 Ascendency

A key contribution of the Ulanowicz School to ENA is Ascendency concept and the development of several information based indices (Ulanowicz, 1986, 1997). This analysis is based on all of the flows in the system and does not assume the modeled system is at steady-state. The `enaAscendency` function returns several of these information based measures.

```
> enaAscendency(oyster)

          AMI      ASC       OH      CAP   ASC.CAP    OH.CAP
[1,] 1.330211 166.3473 211.0979 377.4452 0.4407191 0.5592809
     ASC.OH.RSUM
[1,]           1
```

These network statistics are as follows

- AMI = Average Mutual Information
- ASC = Ascendency
- OH = Overhead
- CAP = Capacity
- ASC.CAP = the Ascendency to Capacity Ratio
- OH.CAP = the Overhead to Capacity Ratio
- ASC.OH.RSUM = sum of Ascendency and Overhead, which should equal 1

6.4 Storage Analysis

Storage ENA was developed in the Patten School. It is similar to flow ENA, but divides by storage (e.g., biomass) instead of throughflow. See Fath and Patten (1999) and Schramski et al. (2011) for an overview of this method. This is again implemented as in NEA.m (Fath and Borrett, 2006), so the resultant matricies are oriented column-to-row.

```
> S <- enaStorage(m)
> attributes(S)

$names
 [1] "C"  "P"  "S"  "Q"  "CP" "PP" "SP" "QP" "dt" "ns"

> S$ns

          TSS       CIS NAS NASP   IDS.i    IDS.o    IDS.r  HMG.S.o
[1,] 3112.044 0.9940252  20   21 454.227 294.1527 299.1171 1.115985
     HMG.S.i     lam1P     rhoP    lam1PP    rhoPP    AGG.S
[1,] 1.38251 0.9975603 1.001521 0.9975603 1.001521 75.04326
```

6.5 Utility Analysis

Utility analysis describes the relationship between node pairs in the ecosystem model when considering both direct and indirect interactions. It developed in the Patten School (Fath and Patten, 1999; Patten, 1991) and is similar to yet distinct from the Ulanowicz School mixed trophic impacts analysis (Ulanowicz and Puccia, 1990). Utility analysis can be conducted from both the flow and storage perspectives, so the "type" argument needs to be set to suit the users needs. This is again implemented as in NEA.m.

```
> UF <- enaUtility(m,eigen.check=TRUE,type="flow")
> US <- enaUtility(m,eigen.check=TRUE,type="storage")
> attributes(UF)

$names
[1] "D"  "U"  "Y"  "ns"

>
```

Please note the function argument "eigen.check=TRUE". For this analysis to work, the power series of the direct utility matrices must converge, which is only true if the dominant eigenvalue of the direct utility matrix is less than 1. The function default prevents the analysis from being performed if this condition is not met. Users that wish to perform the analysis anyway can set "eigen.check=FALSE". Care should be used when doing this, as the meaning of the underlying mathematics is uncertain.

6.6 Environ Analysis

Environ Analysis finds the *n unit* input and *n* output environs for the model (Fath and Patten, 1999; Patten, 1978). These unit environs are returned by the *environ* function as in NEA.m. They indicate the flow activity in each subnetwork generated by pulling a unit out of a node (input environs) or pushing a unit into a node (output environ). These unit environs can be converted into "realized" environs by multiplying each by the relevant observed input or output (Borrett and Freeze, 2011).

```
> E <- environ(m)
> attributes(E)

$names
[1] "input"  "output"

> E$output[1]

$`Filter Feeders`
                         [,1]        [,2]        [,3]         [,4]
Filter Feeders    -1.00000000  0.00000000  0.00000000  0.000000000
Microbiota         0.00000000 -0.19706053  0.00000000  0.000000000
Meiofauna          0.00000000  0.02908126 -0.20449723  0.000000000
Deposit Feeders    0.00000000  0.02908126  0.01593682 -0.060525681
Predators          0.01238245  0.00000000  0.00000000  0.004149988
Deposited Detritus 0.38078129  0.00000000  0.10224982  0.045999518
```

```
                    0.60683627  0.13889800  0.08631059  0.010376176
                           [,5]       [,6] [,7]
Filter Feeders      0.000000000  0.0000000    1
Microbiota          0.000000000  0.1970605    0
Meiofauna           0.000000000  0.1754160    0
Deposit Feeders     0.000000000  0.0155076    0
Predators          -0.016532433  0.0000000    0
Deposited Detritus  0.007865927 -0.5368966    0
                    0.008666506  0.1489125    0
```

The *TET* function returns vectors of the unit and realized input and output total environ throughflow. The realized total environ throughflow is an environ based partition of the total system throughflow (TST).

```
> tet <- TET(m)
> show(tet)

$realized.input
[1] 25.165000 22.647638 14.582798  2.028052  1.053786 18.107007

$realized.output
[1] 83.5833  0.0000  0.0000  0.0000  0.0000  0.0000

$unit.input
[1] 1.000000 3.931882 4.074090 4.713111 2.932069 2.931882

$unit.output
[1] 2.015512 1.836089 2.540670 3.124836 2.234317 2.594261
```

The *TES* functions returns the both the realized and unit total environ storage for the input and output environs. Again, the realized TES is a partition of the total system storage (TSS).

```
> tes <- TES(m)
> show(tes)

$realized.input
    Filter Feeders         Microbiota          Meiofauna
        2000.00000            2.41209           24.12171
   Deposit Feeders          Predators Deposited Detritus
          16.27440           69.23803         1000.03118

$realized.output
[1] 3112.044    0.000    0.000    0.000    0.000    0.000

$unit.input
    Filter Feeders         Microbiota          Meiofauna
       289.3658066          0.6561948          7.3735209
   Deposit Feeders          Predators Deposited Detritus
        11.5308112        109.7205293        265.1036470
```

```
$unit.output
    Filter Feeders         Microbiota          Meiofauna
          75.04326           16.06273           41.03146
   Deposit Feeders          Predators Deposited Detritus
          65.81279          132.44451           66.11575
```

6.7 Control Analysis

Control analysis is implemented as in the original NEA.m function. Recent updates to control analysis (e.g., Schramski et al., 2006, 2007) still need to be included.

```
> C <- enaControl(m)
> attributes(C)

$names
[1] "CN" "CQ"
```

The flow control **CN** and storage control **CQ** matrices are oriented column-to-row.

6.8 Mixed Trophic Impacts

Mixed Trophic Impacts is a popular analysis from the Ulanowicz School of ENA (Ulanowicz and Puccia, 1990). The `mixedTrophicImpacts` function generates comparable results to the calculations in Ulanowicz and Puccia (1990).

```
> mti <- mixedTrophicImpacts(oyster)
> attributes(mti)

$names
[1] "G"  "FP" "Q"  "M"

> mti$M  # shows the total impact matrix

[1] NA
```

In this case, the power series of the direct trophic impacts matrix does not converge (dominant eigenvalue is greater than one). Thus, the function returns the mti\$M = NA. Like with Utility analysis, however, we can use the eigen.check argument to do the calculation despite the mathematical problem.

```
> mti <- mixedTrophicImpacts(oyster,eigen.check=FALSE)
> attributes(mti)

$names
[1] "G"  "FP" "Q"  "M"

> mti$M  # shows the total impact matrix
```

```
                   Filter Feeders  Microbiota    Meiofauna
Filter Feeders      -0.0250635283  0.16956382  0.431493557
Microbiota          -0.0015848556 -0.30675078 -0.182458391
Meiofauna           -0.0001241781 -0.47413204 -0.070959618
Deposit Feeders     -0.0069255188 -0.26769125 -0.007062628
Predators           -0.0301817448  0.02000515 -0.004028911
Deposited Detritus  -0.0034657973  0.21795628  0.612654910
                   Deposit Feeders    Predators Deposited Detritus
Filter Feeders          0.26144106  0.795834137        0.516016759
Microbiota              0.20520368  0.050323410       -0.295378609
Meiofauna               0.01607831  0.003942987       -0.001592286
Deposit Feeders        -0.10329881  0.219903765        0.177109591
Predators              -0.07586335 -0.041648786       -0.019939324
Deposited Detritus      0.44874394  0.110048344       -0.251366300
```

In this case, the **G**, **FP**, **Q**, and **M** matrices are oriented from row-to-column.

6.9 Other Analyses

There are a number of additional tools in the package. Here we highlight a couple of them.

A quick way to get a list of all of the global network statistics reported in Structure, Flow, Ascendency, Storage, and Utility analysis is to use the `get.ns` function.

```
> ns <- get.ns(m)
> str(ns)    # examine the structure of ns
```

```
'data.frame':        1 obs. of  60 variables:
 $ n           : num 6
 $ L           : num 12
 $ C           : num 0.333
 $ LD          : num 2
 $ lam1A       : num 2.15
 $ mlam1A      : num 1
 $ lam2A       : num 1
 $ rho         : num 2.15
 $ R           : num 0.466
 $ d           : num 0.148
 $ no.scc      : num 2
 $ no.scc.big  : num 1
 $ pscc        : num 0.833
 $ Boundary    : num 41.5
 $ TST         : num 83.6
 $ TSTp        : num 125
 $ APL         : num 2.02
 $ FCI         : num 0.11
 $ BFI         : num 0.496
 $ DFI         : num 0.195
 $ IFI         : num 0.309
 $ id          : num 1.58
 $ id.I        : num 1.72
 $ id.O        : num 1.53
 $ HMG.I       : num 2.05
 $ HMG.O       : num 1.89
 $ AMP.strong.I: num 3
 $ AMP.strong.O: num 1
 $ mode0       : num 41.5
 $ mode1       : num 32.9
 $ mode2       : num 9.21
 $ mode3       : num 32.9
 $ mode4       : num 41.5
 $ AMI         : num 1.33
 $ ASC         : num 166
 $ OH          : num 211
 $ CAP         : num 377
 $ ASC.CAP     : num 0.441
 $ OH.CAP      : num 0.559
 $ ASC.OH.RSUM : num 1
 $ TSS         : num 3112
 $ CIS         : num 0.994
 $ NAS         : num 20
 $ NASP        : num 21
 $ IDS.i       : num 454
 $ IDS.o       : num 294
 $ IDS.r       : num 299
 $ HMG.S.o     : num 1.12
 $ HMG.S.i     : num 1.38
 $ lam1P       : num 0.998
 $ rhoP        : num 1
 $ lam1PP      : num 0.998
 $ rhoPP       : num 1
 $ AGG.S       : num 75
 $ lam1D       : num 0.899
 $ synergism.F : num 4.92
 $ mutualism.F : num 2.27
 $ lam1DS      : num 0.302
 $ synergism.S : num 13.1
 $ mutualism.S : num 2.6
```

Centrality analysis is a large topic in network science. Fann and Borrett (2012) introduced an environ based centrality and contrasted it with the more commonly used eigenvector centrality. Both of these centralities can be calculated in *enaR* as follows:

```
> F <- enaFlow(oyster)
> ec <- environCentrality(F$N)
> show(ec)
```

```
$ECin
    Filter Feeders         Microbiota          Meiofauna
```

```
       0.06970737         0.19108709         0.20595483
   Deposit Feeders          Predators Deposited Detritus
       0.12350944         0.07903903         0.33070223

$ECout
    Filter Feeders         Microbiota          Meiofauna
         0.1404961          0.1279889          0.1771034
   Deposit Feeders          Predators Deposited Detritus
         0.2178241          0.1557484          0.1808391

$AEC
    Filter Feeders         Microbiota          Meiofauna
         0.1051017          0.1595380          0.1915291
   Deposit Feeders          Predators Deposited Detritus
         0.1706668          0.1173937          0.2557707

> eigenCentrality(F$G)

$EVCin
[1] 0.00000000 0.23325048 0.26566843 0.11130122 0.01286707 0.37691280

$EVCout
[1] 0.1207568 0.1093625 0.1876329 0.2518905 0.1470501 0.1833072

$AEVC
[1] 0.06037842 0.17130647 0.22665067 0.18159586 0.07995858 0.28011000

>
```

These centrality values have been normalized to sum to one.

Figure 3 shows one way to visualize the Average Environ Centralities.

```
> # set plotting parameters
> opar <- par(las=1,mar=c(7,5,1,1),xpd=TRUE,bg="white")
> # find centrality order
> o <- order(ec$AEC,decreasing=TRUE)
> bp <- barplot(ec$AEC[o],     # create barplot
+               names.arg=NA,
+               ylab="Average Environ Centrality",
+               col="blue",border=NA)
> text(bp,-0.008,                # add labels
+      labels=names(ec$AEC)[o],
+      srt=35,adj=1,cex=1)
> rm(opar)  # remove the plotting parameters
>
```

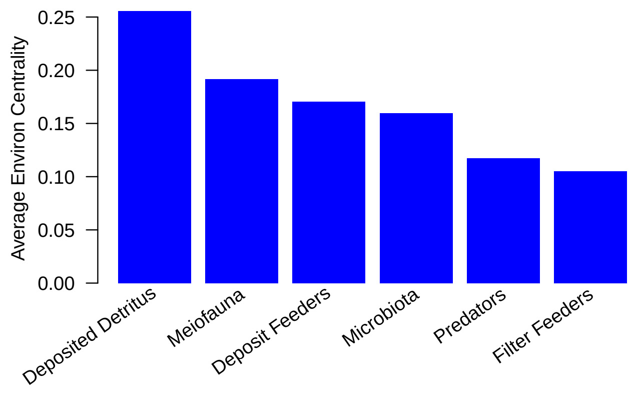

Figure 2: Bar plot of the Oyster Reef model Average Environ Centralities.

7 Multi-Model Analyses (Batch Processing)

While many investigators analyze single models, much of ENA is used to compare ecosystem models (e.g., Baird et al., 1991, 1995; Christian and Thomas, 2003; Whipple et al., 2007). Investigators have also analyzed large set of models to determine the generality of hypothesized ecosystem properties (e.g., Borrett and Salas, 2010; Christensen, 1995; Salas and Borrett, 2011). For both of these applications, investigators need to analyze multiple models. One advantage of the *enaR* R package is that it simplifies this batch processing. Here we illustrate how to batch analyze a selection of models.

Our first step is to read in the model data from multiple SCOR formatted data files.

```
> old.wd <- getwd()
> setwd("~/Dropbox/RENA/package/data/")
> #list of model names to analyze
> models <- c('oyster.dat',
+              'cone_spring.dat',
+              'LakeLanier_avg_N.dat',
+              'Sylt_Romo_Bight_C.dat',
+              'Sylt_Romo_Bight_N.dat',
+              'Sylt_Romo_Bight_P.dat',
+              'chesapeake_mesohaline_C_annual.dat',
+              'chesapeake_meso_ann_N.dat',
+              'chesapeake_meso_ann_P.dat',
+              'mdmar02.bal')
> mm <- length(models) # number of models
> model.list=list()    # initialize the model list
> # loop through models, #storing the network objects in a list.
> for(i in 1:mm){
+    model.list[[i]] <- read.scor(models[i])
+  }
> setwd(old.wd) #return to original working directory
```

Now that we have the raw data loaded, we can start to manipulate it. The first step is to balance the models and then we can run the flow analysis. We are using the `lapply` function to apply the analysis across the list of models stored in model.list.

```
> # balance models as necessary
> m.list <- lapply(model.list,balance)

[1] BALANCED
[1] BALANCED
[1] BALANCED
[1] AVG2
[1] BALANCED
[1] AVG2
[1] BALANCED
[1] BALANCED
[1] AVG2
[1] BALANCED

> # if balancing fails, you can use force.balance
> # to repeatedly apply the balancing procedure
> unlist(lapply(m.list,ssCheck))

 [1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE

> m.list <- lapply(model.list,force.balance)

[1] AVG2
[1] AVG2
[1] AVG2
[1] AVG2
[1] AVG2
[1] AVG2
[1] AVG2
[1] AVG2

> unlist(lapply(m.list,ssCheck))

 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

> # Example Flow Analysis
> F.list <- lapply(m.list, enaFlow)
> # the full results of the flow analysis is now stored in the elements
> # of the F.list.  To get the results for just the first model...
> F.list[[1]]

$T
   Filter Feeders         Microbiota          Meiofauna
          41.4700             8.1721             8.4805
  Deposit Feeders          Predators Deposited Detritus
           2.5100             0.6856            22.2651
```

```
$G
                   Filter Feeders Microbiota  Meiofauna Deposit Feeders
Filter Feeders         0.00000000  0.0000000 0.00000000      0.00000000
Microbiota             0.00000000  0.0000000 0.00000000      0.00000000
Meiofauna              0.00000000  0.1475753 0.00000000      0.00000000
Deposit Feeders        0.00000000  0.1475753 0.07793173      0.00000000
Predators              0.01238245  0.0000000 0.00000000      0.06856574
Deposited Detritus     0.38078129  0.0000000 0.50000590      0.76000000
                   Predators Deposited Detritus
Filter Feeders     0.0000000         0.00000000
Microbiota         0.0000000         0.36703630
Meiofauna          0.0000000         0.32672209
Deposit Feeders    0.0000000         0.02888377
Predators          0.0000000         0.00000000
Deposited Detritus 0.4757876         0.00000000

$GP
                   Filter Feeders Microbiota Meiofauna Deposit Feeders
Filter Feeders          0.0000000  0.0000000 0.0000000      0.00000000
Microbiota              0.0000000  0.0000000 0.0000000      0.00000000
Meiofauna               0.0000000  0.1422086 0.0000000      0.00000000
Deposit Feeders         0.0000000  0.4804781 0.2633068      0.00000000
Predators               0.7489790  0.0000000 0.0000000      0.25102100
Deposited Detritus      0.7092265  0.0000000 0.1904460      0.08567669
                    Predators Deposited Detritus
Filter Feeders     0.00000000          0.0000000
Microbiota         0.00000000          1.0000000
Meiofauna          0.00000000          0.8577914
Deposit Feeders    0.00000000          0.2562151
Predators          0.00000000          0.0000000
Deposited Detritus 0.01465073          0.0000000

$N
                   Filter Feeders Microbiota  Meiofauna Deposit Feeders
Filter Feeders         1.00000000 0.00000000 0.00000000      0.00000000
Microbiota             0.19706053 1.10186299 0.28629883      0.40394538
Meiofauna              0.20449723 0.25328240 1.29710322      0.41918954
Deposit Feeders        0.06052568 0.19036255 0.16586629      1.12406883
Predators              0.01653243 0.01305235 0.01137274      0.07707261
Deposited Detritus     0.53689655 0.27752837 0.78002865      1.10055975
                   Predators Deposited Detritus
Filter Feeders     0.0000000         0.00000000
Microbiota         0.2424763         0.50963134
Meiofauna          0.2516269         0.52886388
Deposit Feeders    0.0744748         0.15652949
Predators          1.0051064         0.01073256
Deposited Detritus 0.6606330         1.38850389
```

```
$NP
                   Filter Feeders Microbiota Meiofauna Deposit Feeders
Filter Feeders                  1  0.0000000 0.0000000       0.0000000
Microbiota                      1  1.1018630 0.2971032       0.1240688
Meiofauna                       1  0.2440716 1.2971032       0.1240688
Deposit Feeders                 1  0.6197856 0.5604100       1.1240688
Predators                       1  0.1555792 0.1406747       0.2821649
Deposited Detritus              1  0.1018630 0.2971032       0.1240688
                   Predators Deposited Detritus
Filter Feeders     0.0000000          0.0000000
Microbiota         0.0203426          1.3885039
Meiofauna          0.0203426          1.3885039
Deposit Feeders    0.0203426          1.3885039
Predators          1.0051064          0.3485436
Deposited Detritus 0.0203426          1.3885039

$ns
     Boundary     TST     TSTp      APL       FCI       BFI       DFI
[1,]    41.47 83.5833 125.0533 2.015512 0.1101686 0.4961517 0.1950689
           IFI       id     id.I     id.O    HMG.I    HMG.O
[1,] 0.3087794 1.582925 1.716607 1.534181 2.051826 1.891638
     AMP.strong.I AMP.strong.O mode0    mode1    mode2    mode3 mode4
[1,]            3            1 41.47 32.90504 9.208256 32.90504 41.47

>
```

We can use the same technique to extract specific information, like just the ratio of Indirect-to-Direct flow for each model.

```
> # Example of extracting just specific information - Indirect Effects Ratio
> IDs <- unlist(lapply(m.list, function(x) enaFlow(x)$ns[8]))
> names(IDs) <- models
> show(IDs)

                          oyster.dat                     cone_spring.dat
                           0.3087794                           0.3105362
                LakeLanier_avg_N.dat             Sylt_Romo_Bight_C.dat
                           0.7694837                           0.3635677
               Sylt_Romo_Bight_N.dat             Sylt_Romo_Bight_P.dat
                           0.5942724                           0.9131001
chesapeake_mesohaline_C_annual.dat         chesapeake_meso_ann_N.dat
                           0.5320786                           0.7136606
           chesapeake_meso_ann_P.dat                        mdmar02.bal
                           0.8191133                           0.3735556

>
```

We can also collect the set of output-oriented integral flow matrices.

```
> # Here is a list containing only the output-oriented integral flow matrices
> N.list <- lapply(m.list,function(x) enaFlow(x)$N)
```

We can also apply the `get.ns` function to extract all of the network statistics for each model. We then use the `do.call` function to reshape the network statistics into a single data frame.

```
> # Collecting and combining all network statistics
> ns.list <- lapply(m.list,get.ns) # returns as list
> ns <- do.call(rbind,ns.list)  # ns as a data.frame
> rownames(ns) <- models
> str(ns)
```

```
'data.frame':        10 obs. of  60 variables:
 $ n           : num  6 5 11 59 59 59 36 36 36 49
 $ L           : num  12 8 26 276 328 328 122 153 153 386
 $ C           : num  0.3333 0.32 0.2149 0.0793 0.0942 ...
 $ LD          : num  2 1.6 2.36 4.68 5.56 ...
 $ lam1A       : num  2.15 1.84 2.75 6.72 7.2 ...
 $ mlam1A      : num  1 1 1 1 1 1 1 1 1 1
 $ lam2A       : num  1 1 1.28 3.4 3.85 ...
 $ rho         : num  2.15 1.84 2.14 1.98 1.87 ...
 $ R           : num  4.66e-01 6.69e-01 1.85e-01 0.00 4.37e-32 ...
 $ d           : num  0.148 0.239 0.382 2.042 1.645 ...
 $ no.scc      : num  2 2 1 11 1 1 16 1 1 3
 $ no.scc.big  : num  1 1 1 1 1 1 2 1 1 1
 $ pscc        : num  0.833 0.8 1 0.831 1 ...
 $ Boundary    : num  41.5 11819 95.8 683448.3 99613 ...
 $ TST         : num  8.36e+01 3.06e+04 7.49e+02 1.78e+06 3.64e+05 ...
 $ TSTp        : num  125 42445 845 2106101 463305 ...
 $ APL         : num  2.02 2.59 7.82 2.61 3.65 ...
 $ FCI         : num  0.1102 0.0919 0.3988 0.0943 0.2314 ...
 $ BFI         : num  0.496 0.386 0.128 0.384 0.274 ...
 $ DFI         : num  0.195 0.304 0.103 0.253 0.132 ...
 $ IFI         : num  0.309 0.311 0.769 0.364 0.594 ...
 $ id          : num  1.58 1.02 7.5 1.44 4.51 ...
 $ id.I        : num  1.72 1.41 7.07 1.47 4.21 ...
 $ id.O        : num  1.534 0.913 7.222 1.133 2.945 ...
 $ HMG.I       : num  2.05 2.47 3.15 1.76 2.15 ...
 $ HMG.O       : num  1.89 1.87 3.29 1.72 2.11 ...
 $ AMP.strong.I: num  3 4 22 8 88 365 31 83 144 69
 $ AMP.strong.O: num  1 0 18 9 59 323 11 58 85 22
 $ mode0       : num  41.5 11819 95.8 683448.3 99613 ...
 $ mode1       : num  32.9 15991.3 354.7 929687.6 179907.2 ...
 $ mode2       : num  9.21 2.82e+03 2.99e+02 1.68e+05 8.42e+04 ...
 $ mode3       : num  32.9 15991.3 354.7 929687.6 179907.2 ...
 $ mode4       : num  41.5 11819 95.8 683448.3 99613 ...
 $ AMI         : num  1.33 1.34 1.87 1.9 1.89 ...
 $ ASC         : num  166 56725 1579 4686962 876030 ...
 $ OH          : num  211 79139 2581 7333395 1167766 ...
 $ CAP         : num  377 135865 4160 12020357 2043795 ...
 $ ASC.CAP     : num  0.441 0.418 0.38 0.39 0.429 ...
 $ OH.CAP      : num  0.559 0.582 0.62 0.61 0.571 ...
 $ ASC.OH.RSUM : num  1 1 1 1 1 1 1 1 1 1
 $ TSS         : num  3112 5 1634 53347 10987 ...
 $ CIS         : num  0.994 0.516 0.862 0.989 0.997 ...
 $ NAS         : num  20 4 91 377 1443 ...
 $ NASP        : num  21 4 83 532 1481 ...
 $ IDS.i       : num  454.2 12.1 39.3 391.5 2087.4 ...
 $ IDS.o       : num  294.2 12.1 36.1 274 1602.1 ...
 $ IDS.r       : num  299.12 3.87 34.05 262.38 1688.48 ...
 $ HMG.S.o     : num  1.116 0.681 2.239 1.014 1.495 ...
 $ HMG.S.i     : num  1.383 0.681 2.25 1.308 1.704 ...
 $ lam1P       : num  0.998 0.968 0.978 0.999 1 ...
 $ rhoP        : num  1 1.21 1.06 1 1 ...
 $ lam1PP      : num  0.998 0.968 0.978 0.999 1 ...
 $ rhoPP       : num  1 1.21 1.06 1 1 ...
 $ AGG.S       : num  7.50e+01 4.23e-04 1.71e+01 7.81e-02 1.10e-01 ...
 $ lam1D       : num  0.899 1.016 1.213 1.264 1.282 ...
 $ synergism.F : num  4.92 3.98 3.8 3.21 3.92 ...
 $ mutualism.F : num  2.273 2.125 2.184 0.774 0.847 ...
 $ lam1DS      : num  0.302 9815.89 0.667 751.549 2196.043 ...
 $ synergism.S : num  13.09 1031.213 4.462 0.944 6.789 ...
 $ mutualism.S : num  2.6 3.17 1.75 1.05 1.46 ...
```

Given this data frame of network statistics, we can construct interesting plots for further analysis.

```
> #quartz("Quartz",width=8,height=4.5,pointsize=12)
> opar <- par(las=1,mar=c(8,5,2,1),xpd=TRUE,mfrow=c(1,2))
> bp=barplot(ns$id,ylab="Indirect-to-Direct Flow Ratio (I/D, Realized)",
+          col="darkgreen",border=NA)
> text(bp,-1,                 # add labels
+       labels=rownames(ns),
+         srt=35,adj=1,cex=0.85)
> opar <- par(xpd=FALSE)
> abline(h=1,col="orange",lwd=2)
> #
> plot(ns$FCI,ns$id,pch=20,col="blue",cex=2,
+       ylab="Indirect-to-Direct Flow Ratio (I/D, Realized)",
+       xlab="Finn Cycling Index (FCI)",
+       xlim=c(0,0.8),ylim=c(0,22))
> #
> rm(opar)  # remove the plotting parameters
>
```

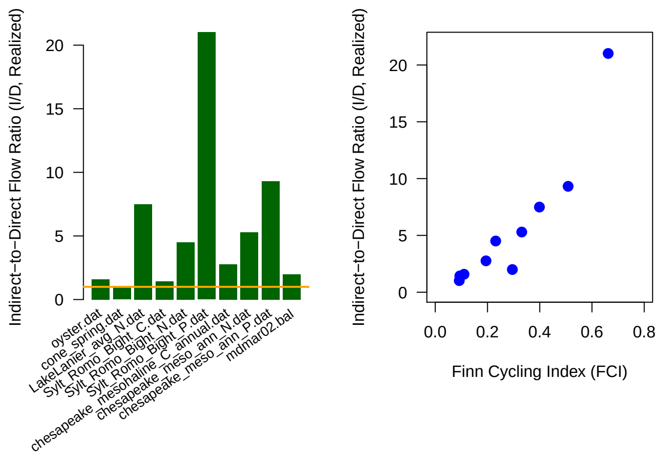

Figure 3: Ratio of Indirect-to-Direct Flow for nine ecosystem models (left) and relationship between the Finn Cycling Index and the ratio of Indirect-to-Direct flow in the nine models.

8 Connecting to Other Useful Packages

Another advantage of building the *enaR* package in R is that it lets ecologists take advantage of other types of network analysis and statistical tools that already exist in R. We highlight two examples here.

8.1 sna: Social Network Analysis

The *sna* package for Social Network Analysis is bundled in the *statnet* package and uses the same network data object defined in *network* that we selected to use for *enaR*. Thus, the design decision to use the network data object gives users direct access to *sna* tools.

Multiple measures of network centrality have been proposed, and the *sna* package provides a way of calculating several. Thus, ecologists can now use the sna algorithms to determine different types of centrality for their models.

```
> betweenness(oyster)

[1] 0.0 0.0 0.5 3.5 0.0 9.0

> closeness(oyster)

[1] 0.625 0.000 0.000 0.000 0.000 0.000
```

The *sna* package introduced new graphical capabilities as well. For example, it will create a target diagram of centralities.

```
> m <- read.scor("~/Dropbox/RENA/package/pkg_skeleton/ena/data/chesapeake_mesohaline_C_annual.
> b <- betweenness(m)         # calculate betweenness centrality
> nms <- m%v%'vertex.names'   # get vertex names
> show(nms)

 [1] "Phytoplankton"                  "Bacteria in Suspended POC"
 [3] "Bacteria in Sediment POC"       "Benthic Diatoms"
 [5] "Free Bacteria"                  "Heterotrophic Microflagelates"
 [7] "Ciliates"                       "Zooplankton"
 [9] "Ctenophores"                    "Sea Nettle"
[11] "Other Suspension Feeders"       "Mya arenaria"
[13] "Oysters"                        "Other Polychaetes"
[15] "Nereis"                         "Macoma spp."
[17] "Meiofauna"                      "Crustacean Deposit Feeder"
[19] "Blue Crab"                      "Fish Larvae"
[21] "Alewife & Blue Herring"         "Bay Anchovy"
[23] "Menhaden"                       "Shad"
[25] "Croaker"                        "Hogchoker"
[27] "Spot"                           "White Perch"
[29] "Catfish"                        "Bluefish"
[31] "Weakfish"                       "Summer Flounder"
[33] "Striped Bass"                   "Dissolved Organic Carbon"
[35] "Suspended Particulate Carbon"  "Sediment Partculate Carbon"

> nms[b<=(0.1*max(b))] <- NA  # exclude less central nodes
> set.seed(3)
> opar <- par(xpd=TRUE,mfrow=c(1,1))
> # create target plot
> gplot.target(m,b,#circ.lab=FALSE,
+               edge.col="grey",
+               label=nms) # show only labels of most central nodes
>              #xlim=c(-1,4))
> rm(opar)
```

In addition to the node-level measures, *sna* includes graph-level indices.

```
> centralization(oyster, degree)

[1] 0.45

> centralization(oyster,closeness)

[1] 0.75

> centralization(oyster,betweenness)

[1] 0.41

>
```

```
 [1] "Phytoplankton"                    "Bacteria in Suspended POC"
 [3] "Bacteria in Sediment POC"         "Benthic Diatoms"
 [5] "Free Bacteria"                    "Heterotrophic Microflagelates"
 [7] "Ciliates"                         "Zooplankton"
 [9] "Ctenophores"                      "Sea Nettle"
[11] "Other Suspension Feeders"         "Mya arenaria"
[13] "Oysters"                          "Other Polychaetes"
[15] "Nereis"                           "Macoma spp."
[17] "Meiofauna"                        "Crustacean Deposit Feeder"
[19] "Blue Crab"                        "Fish Larvae"
[21] "Alewife & Blue Herring"           "Bay Anchovy"
[23] "Menhaden"                         "Shad"
[25] "Croaker"                          "Hogchoker"
[27] "Spot"                             "White Perch"
[29] "Catfish"                          "Bluefish"
[31] "Weakfish"                         "Summer Flounder"
[33] "Striped Bass"                     "Dissolved Organic Carbon"
[35] "Suspended Particulate Carbon"     "Sediment Partculate Carbon"
```

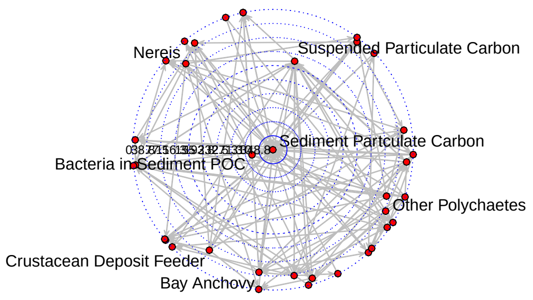

Figure 4: Target plot of node betweenness centrality for the Chesapeake Bay model (mesohaline, carbon, annual).

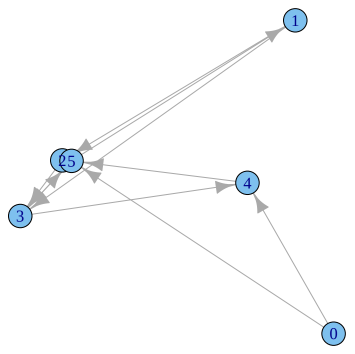

Figure 5: Plot of Oyster reef model using *iGraph*

8.2 iGraph

The *iGraph* package can also be useful for analyzing network data. Here are a few examples of using the package. Note that some functions in *iGraph* conflict with other functions already defined, so care is required when using *iGraph*.

```
> library(igraph)
> # The adjacency matrix
> A <- t(St$A)  # I am transposing the matrix to be
>               # row-to-column oriented as expected by iGraph
>
> # creating an iGraph graph
> g <- graph.adjacency(A)
> plot(g)  # uses iGraph plot tools
```

iGraph has a different set of visualization tools and generates a different looking graph (Fig. 5).

```
> # betweenness centrality (calculated by iGraph and sna)
> betweenness(g)

[1] 0.0 0.0 0.5 3.5 0.0 9.0

> # shortest path between any two nodes
> shortest.paths(g)

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    0    2    2    2    1    1
[2,]    2    0    1    1    2    1
[3,]    2    1    0    1    2    1
[4,]    2    1    1    0    1    1
[5,]    1    2    2    1    0    1
[6,]    1    1    1    1    1    0
```

```
> # average path length in the network (graph theory sense)
> average.path.length(g,directed=TRUE)

[1] 1.52

> diameter(g)  # diameter of the graph

[1] 2

> vertex.connectivity(g)  # connectivity of a graph (group cohesion)

[1] 0

> subcomponent(g,1,'in')  # subcomponent reachable from 1 along inputs

[1] 1 5 0 2 3 4

> subcomponent(g,2,'in')  # subcomponent reachable from 2 along inputs

[1] 2 1 5 0 3 4

> subcomponent(g,1,'out') # subcomponent reachable from 1 along outputs

[1] 1 2 3 5 4

> subcomponent(g,2,'out') # subcomponent reachable from 2 along output

[1] 2 3 5 4 1

> edge.connectivity(g)

[1] 0

> detach(package:igraph)  # detach igraph package
```

There are other R packages that have graph and network analysis tools, like Bioconductor, that might also be useful for ecologists

9 Summary and Future

This vignette shows how to use several of the key features of the *enaR* package that enables scientists to perform Ecological Network Analysis in R. The vision for this package is that it will provide access to ENA algorithms from both the Ulanowicz and Patten Schools. In its current form it replicates, updates, and extends the functionality of the NEA.m function (Fath and Borrett, 2006). It also includes both ascendency calculations and mixed trophic impacts from the Ulanowicz school of ENA, but there remains many possibilities for future development. We hope to do this in collaboration with users. This vignette also illustrates how users can further analyze their data with other R packages for graph and network analysis like *sna* and *iGraph.* In summary, we hope you find this package useful for your ENA needs.

References

Allesina, S., A. Bodini, and C. Bondavalli. 2005. Ecological subsystems via graph theory: the role of strongly connected components. Oikos **110**:164–176.

Allesina, S., and C. Bondavalli. 2003. Steady state of ecosystem flow networks: A comparison between balancing procedures. Ecol. Model. **165**:221–229.

Allesina, S., and C. Bondavalli. 2004. WAND: An ecological network analysis user-friendly tool. Environ. Model. Softw. **19**:337–340.

Baird, D., J. M. McGlade, and R. E. Ulanowicz. 1991. The comparative ecology of six marine ecosystems. Philos. Trans. R. Soc. Lond. B **333**:15–29.

Baird, D., R. E. Ulanowicz, and W. R. Boynton. 1995. Seasonal nitrogen dynamics in Chesapeake Bay—a network approach. Estuar. Coast. Shelf Sci. **41**:137–162.

Borrett, S. R., R. R. Christian, and R. E. Ulanowicz, 2012. Network Ecology. *in* A. H. El-Shaarawi and W. W. Piegorsch, editors. Encyclopedia of Environmetrics. John Wiley & Sons, 2nd edition.

Borrett, S. R., B. D. Fath, and B. C. Patten. 2007. Functional integration of ecological networks through pathway proliferation. J. Theor. Biol. **245**:98–111.

Borrett, S. R., and M. A. Freeze. 2011. Reconnecting Environs to their Environment. Ecol. Model. **222**:2393–2403.

Borrett, S. R., M. A. Freeze, and A. Salas. 2011. Equivalence of the realized input and output oriented indirect effects metrics in ecological network analysis. Ecol. Model. **222**:2142–2148.

Borrett, S. R., and A. K. Salas. 2010. Evidence for resource homogenization in 50 trophic ecosystem networks. Ecol. Model. **221**:1710–1716.

Borrett, S. R., S. J. Whipple, B. C. Patten, and R. R. Christian. 2006. Indirect effects and distributed control in ecosystems 3. Temporal variability of indirect effects in a seven-compartment model of nitrogen flow in the Neuse River Estuary (USA)—Time series analysis. Ecol. Model. **194**:178–188.

Butts, C. 2008*a*. network: A Package for Managing Relational Data in R. J. Stat. Softw. **24**.

Butts, C. 2008*b*. Social network analysis with sna. J. Stat. Softw. **24**:1–51.

Christensen, V. 1995. Ecosystem maturity—Towards quantification. Ecol. Model. **77**:3–32.

Christensen, V., and C. J. Walters. 2004. Ecopath with Ecosim: Methods, capabilities and limitations. Ecol. Model. **172**:109–139.

Christian, R. R., and C. R. Thomas. 2003. Network analysis of nitrogen inputs and cycling in the Neuse River Estuary, North Carolina, USA. Estuaries **26**:815–828.

Dame, R. F., and B. C. Patten. 1981. Analysis of energy flows in an intertidal oyster reef. Mar. Ecol. Prog. Ser. **5**:115–124.

Fann, S. L., and S. R. Borrett. 2012. Environ centrality reveals the tendency of indirect effects to homogenize the functional importance of species in ecosystems. J. Theor. Biol. **294**:74–86.

Farkas, I., I. Derenyi, A. Barabasi, and T. Vicsek. 2001. Spectra of “real-world” graphs: Beyond the semicircle law. Physical Review E **64**:026704.

Fath, B. D., and S. R. Borrett. 2006. A Matlab© function for Network Environ Analysis. Environ. Model. Softw. **21**:375–405.

Fath, B. D., and B. C. Patten. 1999. Review of the foundations of network environ analysis. Ecosystems **2**:167–179.

Finn, J. T. 1976. Measures of ecosystem structure and function derived from analysis of flows. J. Theor. Biol. **56**:363–380.

Finn, J. T. 1980. Flow analysis of models of the Hubbard Brook ecosystem. Ecology **61**:562–571.

Handcock, M., D. Hunter, C. Butts, S. Goodreau, and M. Morris. 2008. statnet: Software tools for the representation, visualization, analysis and simulation of network data. J. Stat. Softw. **24**:1548.

Hannon, B. 1973. The structure of ecosystems. J. Theor. Biol. **41**:535–546.

Leontief, W. 1936. Quantitative input and output relations in the economic systems of the United States. The Review of Economics and Statistics **18**:105–125.

Leontief, W. W. 1966. Input–Output Economics. Oxford University Press, New York.

Patten, B. C. 1978. Systems approach to the concept of environment. Ohio J. Sci. **78**:206–222.

Patten, B. C., 1991. Network ecology: Indirect determination of the life–environment relationship in ecosystems. Pages 288–351 *in* M. Higashi and T. Burns, editors. Theoretical Studies of Ecosystems: The Network Perspective. Cambridge University Press, New York.

Patten, B. C., R. W. Bosserman, J. T. Finn, and W. G. Cale, 1976. Propagation of cause in ecosystems. Pages 457–579 *in* B. C. Patten, editor. Systems Analysis and Simulation in Ecology, Vol. IV. Academic Press, New York.

Salas, A. K., and S. R. Borrett. 2011. Evidence for dominance of indirect effects in 50 trophic ecosystem networks. Ecol. Model. **222**:1192–1204.

Scharler, U. 2012. Ecosystem development during open and closed phases of temporarily open/closed estuaries on the subtropical east coast of South Africa. Estuar. Coast. Shelf Sci. **108**:119–131.

Scharler, U., and B. Fath. 2009. Comparing network analysis methodologies for consumer–resource relations at species and ecosystems scales. Ecol. Model. **220**:3210–3218.

Schramski, J. R., D. K. Gattie, B. C. Patten, S. R. Borrett, B. D. Fath, C. R. Thomas, and S. J. Whipple. 2006. Indirect effects and distributed control in ecosystems: Distributed control in the environ networks of a seven-compartment model of nitrogen flow in the Neuse River Estuary, USA—Steady-state analysis. Ecol. Model. **194**:189–201.

Schramski, J. R., D. K. Gattie, B. C. Patten, S. R. Borrett, B. D. Fath, and S. J. Whipple. 2007. Indirect effects and distributed control in ecosystems: Distributed control in the environ networks of a seven-compartment model of nitrogen flow in the Neuse River Estuary, USA—Time series analysis. Ecol. Model. **206**:18–30.

Schramski, J. R., C. Kazanci, and E. W. Tollner. 2011. Network environ theory, simulation and EcoNet© 2.0. Environ. Model. Softw. **26**:419–428.

Ulanowicz, R. E. 1986. Growth and Development: Ecosystems Phenomenology. Springer–Verlag, New York.

Ulanowicz, R. E. 1997. Ecology, the Ascendent Perspective. Columbia University Press, New York.

Ulanowicz, R. E. 2004. Quantitative methods for ecological network analysis. Comput. Biol. Chem. **28**:321–339. URL `http://dx.doi.org/10.1016/j.compbiolchem.2004.09.001`.

Ulanowicz, R. E., and J. Kay. 1991. A package for the analysis of ecosystem flow networks. Environmental Software **6**:131–142.

Ulanowicz, R. E., and C. J. Puccia. 1990. Mixed trophic impacts in ecosystems. Coenoses **5**:7–16.

Whipple, S. J., S. R. Borrett, B. C. Patten, D. K. Gattie, J. R. Schramski, and S. A. Bata. 2007. Indirect effects and distributed control in ecosystems: Comparative network environ analysis of a seven-compartment model of nitrogen flow in the Neuse River Estuary, USA—Time series analysis. Ecol. Model. **206**:1–17.